Prelab 11: Pygame

Pygame is a library dedicated to make games using python, on the other hand the graphics library used in lab5 and in project2 was not designed to make games, in fact it was built to help you understand classes and objects and introduce you to programming in Python. Checkout the games made by Pygame: http://www.pygame.org/hifi.html

Pygame installation

First thing to do is to install Pygame on your machine.

For Mac

Open up a terminal:

1. Install homebrew:

   ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

2. Install mercurial

   brew install mercurial

3. Install pygame

   pip3 install hg+http://bitbucket.org/pygame/pygame

For Windows

First, you want to make sure you also have the correct version of Python 3.4 (32-bit, 64-bit) installed, so that if you have a 32-bit windows you should install the 32-bit python and get the 32-bit Pygame and if you have a 64-bit windows you should install the 64-bit python and get the 64-bit Pygame. (You can check your windows version by a right click on “Computer” and click Properties.)
After you install the correct Python version, now download the correct version of Pygame. Go to this page: http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame

Install one of the following: (According to whether you have a 32-bit or 64-bit Windows.)

- pygame-1.9.2a0-cp34-none-win32.whl
- pygame-1.9.2a0-cp34-none-win_amd64.whl

After downloading the file, move it into this directory “C:/Python34/Scripts/”.

Open up a terminal. (Windows + R) and write “cmd” and enter. then run the following commands:

```bash
cd C:\Python34\Scripts
pip.exe install "pygame-1.9.2a0-cp34-none-win32.whl" (or the other one the 64-bit version.)
```

You should see something like this:
Introduction to Pygame

Now your installation is complete, let's make some games!

First game - Keyboard movement

This section will introduce you to Pygame through a simple game. Remember the rectangle bob from project 2, we will develop bob using Pygame, but this time you get to move bob around using the arrows in the keyboard!

Few points to note in the code:

- **RBG model** is a computer science model to define colors. RGB is short for Red Green Blue. Since all colors can be obtained through mixing the primary colors (red, green, blue) in certain amounts, this is how we define colors in computer science as well. When we define a color to be (255,20,147) this means the the red color is mixed with amount 255, and green is 20 and blue is 147. (in fact pink is (255,20,147)). The maximum color value for either red, green or blue is 255 and the minimum is 0. For more information you can check out: [https://en.wikipedia.org/wiki/RGB_color_model](https://en.wikipedia.org/wiki/RGB_color_model).

- **Events and triggers** an event in a program is any change to the state of the inputs of a certain program. In other words, any key pressed on the keyboard is an event, a mouse click is an event as well. Programs usually need to take certain actions according to the type of event. For example a program may want to execute a certain function f1 when the key 'W' is pressed, and want to execute another function f2 when the key 'D' is pressed and a third different function f3 with the left button of the mouse is clicked, this means that an event triggers the execution of a certain code (function) this function called a handler. In our example the event of pressing 'W' triggered the execution of f1 (which is the handler of the event press 'W').

- **The coordinate system** in Pygame is similar to the graphics library (without setting the coordinates). If we created a window(width=800, height=600), then the top left corner point is (0,0), top right (800,0) bottom left (0,600) and bottom right is (800,600).

```python
import pygame # importing the pygame library

# some initializations
pygame.init() # this line initializes pygame
window = pygame.display.set_mode((800,600)) # Create a window with width=800 and height=600
pygame.display.set_caption('Rectangle move') # Change the window's name we create to "Rectangle move"
clock = pygame.time.Clock() # Clocks are used to track and control the frame-rate of a game (how fast and how slow the pace of the game)
    # This line creates and initializes a clock.
```
# color definitions, using RBG color model.
black = (0,0,0)
white = (255,255,255)

# initial center position for the square (bob)
x, y = 0,0

game_loop = True
while game_loop: # this loop will keep going on forever until you quit.
    (close the window)
    for event in pygame.event.get(): # loop through all events
        if event.type == pygame.QUIT: # the event pygame.QUIT is triggered when the user attempts to close the window
            game_loop = False # change the game_loop boolean to False to quit.
        if event.type == pygame.KEYDOWN: # the event pygame.KEYDOWN is triggered when any key on the keyboard is pressed.
            if event.key == pygame.K_LEFT: #If the left arrow key is pressed, then the even pygame.K_LEFT is triggered.
                x -= 50 # move bob 50 to the left
            if event.key == pygame.K_RIGHT: #If the right arrow key is pressed, then the even pygame.K_RIGHT is triggered.
                x += 50 # move bob 50 to the right
            if event.key == pygame.K_DOWN: #If the down arrow key is pressed, then the even pygame.K_DOWN is triggered.
                y += 50 # move bob 50 down
            if event.key == pygame.K_UP: #If the up arrow key is pressed, then the even pygame.K_UP is triggered.
                y -= 50 # move bob 50 up

    # draw and update screen
    window.fill( black ) # fill the screen with black overwriting even bob.
    pygame.draw.rect( window, white, (x, y, 50, 50) ) # draw bob on the screen with new coordinates after its movement.

    # the parameters are as follows: window: is the window object you want to draw on. white: the object color used to fill the rectangle
    # (x,y,50,50) x is the x position of the left side of the rectangle. y is the y position of the upper side of the rectangle.
    # In other words (x,y) is the coordinate of the top left point of the rectangle.
    # 50 is the width,
    # and 50 is the height
    pygame.display.update() #updates the screen with the new drawing of the rectangle.

    #fps stuff:
    clock.tick(10) # this controls the speed of the game. low values makes the game slower, and large values makes the game faster.
Placing shapes on the screen

In this example we will draw a list of rectangles taking a certain formation. A formation is like soccer, each player has a position with respect to other players. In this figure the rectangles are drawn such that each rectangle has a certain position with respect to other rectangles. This program plots n * n squares on the screen, perfectly positioned in the center (like a matrix), the margin between squares and the margin between the screen border should be same. Let the side length of a square is 's' then the margin length is 's', if we have 'n' squares on the same line then we have 'n+1' margins. If the length of the window is 'w', then s = w/(s+s+1)

```
import pygame

WIDTH = 600  # width and height always the same for simplicity
```
n = 3 # the number of squares on the same line.

# some initializations
pygame.init()
window = pygame.display.set_mode((WIDTH, WIDTH))
pygame.display.set_caption('Rectangle matrix')
clock = pygame.time.Clock()

black = (0,0,0)
white = (255,255,255)

length = WIDTH / (2 * n + 1) # the side length of the square, which equals to the length of the margin

game_loop = True
while game_loop:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            game_loop = False

    # draw and update screen
    window.fill(black)
    for i in range(n):
        for j in range(n):
            pygame.draw.rect(window, white, ((i*2+1) * length, (j*2+1) * length, length, length))

    pygame.display.update()

    # fps stuff:
    clock.tick(10)

pygame.quit()